

SOFTWARE ENGINEERING

Staffing Level Estimation and Scheduling

Staffing level estimation

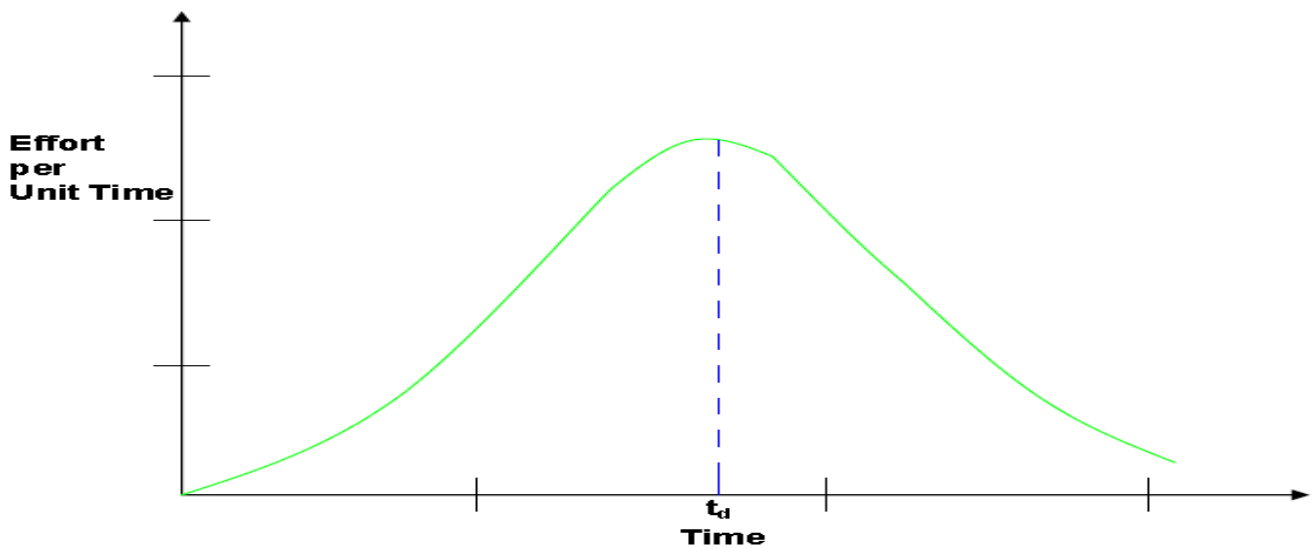
Once the effort required to develop a software has been determined, it is necessary to determine the staffing requirement for the project. Putnam first studied the problem of what should be a proper staffing pattern for software projects. He extended the work of Norden who had earlier investigated the staffing pattern of research and development (R&D) type of projects. In order to appreciate the staffing pattern of software projects, Norden's and Putnam's results must be understood.

Norden's Work

Norden studied the staffing patterns of several R & D projects. He found that the staffing pattern can be approximated by the Rayleigh distribution curve as shown in the figure. Norden represented the Rayleigh curve by the following equation:

$$E = K/t_d^2 * t * e^{-t^2 / 2 t_d^2}$$

Where **E** is the effort required at time **t**. **E** is an indication of the number of engineers (or the staffing level) at any particular time during the duration of the project, **K** is the area under the curve, and **t_d** is the time at which the curve attains its maximum value. It must be remembered that the results of Norden are applicable to general **R & D** projects and were not meant to model the staffing pattern of software development projects.



Rayleigh curve

Putnam's Work

Putnam studied the problem of staffing of software projects and found that the software development has characteristics very similar to other R & D projects studied by Norden and that the Rayleigh-Norden curve can be used to relate the number of delivered lines of code to the effort and the time required to develop the project. By analyzing a large number of army projects, Putnam derived the following expression:

$$L = C_k K^{1/3} t_d^{4/3}$$

The various terms of this expression are as follows:

- **K** is the total effort expended (in PM) in the product development and L is the product size in KLOC.
- **t_d** corresponds to the time of system and integration testing. Therefore, t_d can be approximately considered as the time required to develop the software.
- **C_k** is the state of technology constant and reflects constraints that impede the progress of the programmer. Typical values of **C_k = 2** for poor development environment (no methodology, poor documentation, and review, etc.), **C_k = 8** for good software development environment (software engineering principles are adhered to), **C_k = 11** for an excellent environment (in addition to following software engineering principles, automated tools and techniques are used). The exact value of **C_k** for a specific project can be computed from the historical data of the organization developing it.

Putnam suggested that optimal staff build-up on a project should follow the Rayleigh curve. Only a small number of engineers are needed at the beginning of a project to carry out planning and specification tasks. As the project progresses and more detailed work is required, the number of engineers reaches a peak. After implementation and unit testing, the number of project staff falls.

However, the staff build-up should not be carried out in large installments. The team size should either be increased or decreased slowly whenever required to match the Rayleigh-Norden curve. Experience shows that a very rapid build up of project staff any time during the project development correlates with schedule slippage.

It should be clear that a constant level of manpower through out the project duration would lead to wastage of effort and increase the time and effort required to develop the product. If a constant number of engineers are used over all the phases of a project, some phases would be overstaffed and the other phases would be understaffed causing inefficient use of manpower, leading to schedule slippage and increase in cost.

Effect of schedule change on cost

By analyzing a large number of army projects, Putnam derived the following expression:

$$L = C_k K^{1/3} t_d^{4/3}$$

Where, K is the total effort expended (in PM) in the product development and L is the product size in KLOC, t_d corresponds to the time of system and integration testing and C_k is the state of technology constant and reflects constraints that impede the progress of the programmer

Now by using the above expression it is obtained that,

$$K = L^3 / C_k^3 t_d^4$$

Or,

$$K = C / t_d^4$$

For the same product size, **C = L³ / C_k³** is a constant.

or, $K_1 / K_2 = t_{d2}^4 / t_{d1}^4$

or, $K \propto 1 / t_d^4$

or, **cost** $\propto 1 / t_d$

(as project development effort is equally proportional to project development cost)

From the above expression, it can be easily observed that when the schedule of a project is compressed, the required development effort as well as project development cost increases in proportion to the fourth power of the degree of compression. It means that a relatively small compression in delivery schedule can result in substantial penalty of human effort as well as development cost. For example, if the estimated development time is 1 year, then in order to develop the product in 6 months, the total effort required to develop the product (and hence the project cost) increases 16 times.

Project scheduling

Project-task scheduling is an important project planning activity. It involves deciding which tasks would be taken up when. In order to schedule the project activities, a software project manager needs to do the following:

1. Identify all the tasks needed to complete the project.
2. Break down large tasks into small activities.
3. Determine the dependency among different activities.
4. Establish the most likely estimates for the time durations necessary to complete the activities.
5. Allocate resources to activities.
6. Plan the starting and ending dates for various activities.
7. Determine the critical path. A critical path is the chain of activities that determines the duration of the project.

The first step in scheduling a software project involves identifying all the tasks necessary to complete the project. A good knowledge of the intricacies of the project and the development process helps the managers to effectively identify the important tasks of the project. Next, the large tasks are broken down into a logical set of small activities which would be assigned to different engineers. The work breakdown structure formalism helps the manager to breakdown the tasks systematically.

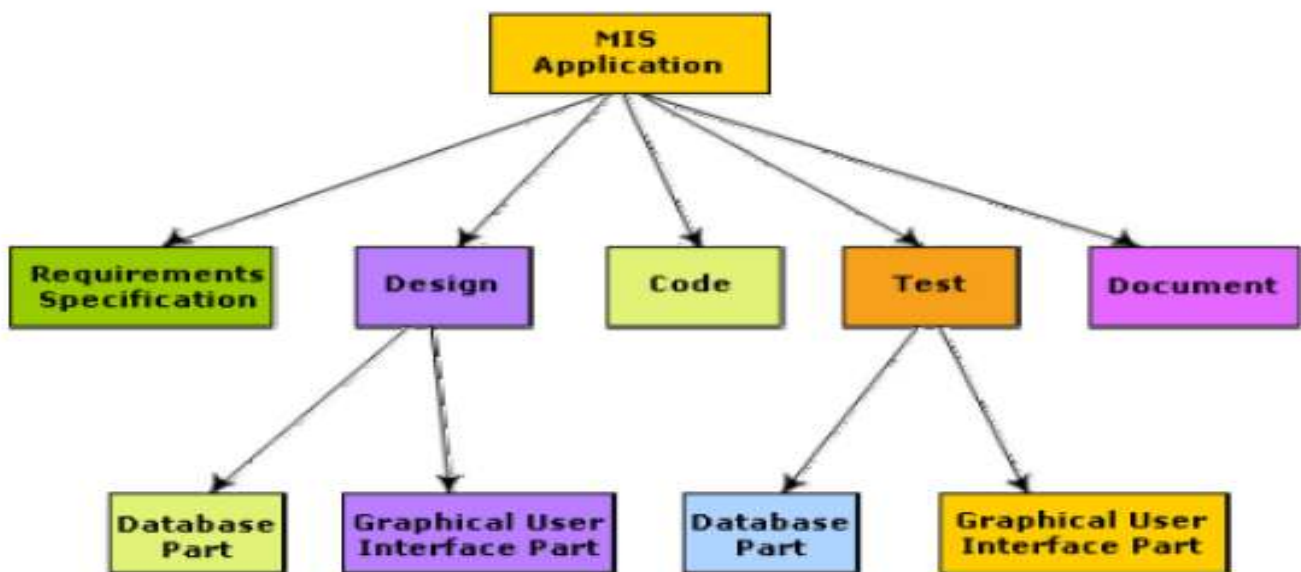
After the project manager has broken down the tasks and created the work breakdown structure, he has to find the dependency among the activities. Dependency among the different activities determines the order in which the different activities would be carried out. If an activity A requires the results of another activity B, then activity A must be scheduled after activity B. In general, the task dependencies define a partial ordering among tasks, i.e. each tasks may precede a subset of other tasks, but some tasks might not have any precedence ordering defined between them (called concurrent task). The dependency among the activities are represented in the form of an activity network.

Once the activity network representation has been worked out, resources are allocated to each activity. Resource allocation is typically done using a Gantt chart. After resource allocation is done, a PERT chart representation is developed. The PERT chart representation is suitable for program monitoring and control. For task scheduling, the project manager needs to decompose the project tasks into a set of activities. The time frame when each activity is to be performed is to be determined. The end of each activity is called milestone. The project manager tracks the progress of a project by monitoring the timely completion of the milestones. If he observes that the milestones start getting delayed, then he has to carefully control the activities, so that the overall deadline can still be met.

Work breakdown structure

Work Breakdown Structure (WBS) is used to decompose a given task set recursively into small activities. WBS provides a notation for representing the major tasks need to be carried out in order to solve a problem. The root of the tree is labeled by the problem name. Each node of the tree is broken down into smaller activities that are made the children of the node. Each activity is recursively decomposed into smaller sub-activities until at the leaf level, the activities requires approximately two weeks to develop. The figure below represents the WBS of an MIS (Management Information System) software.

While breaking down a task into smaller tasks, the manager has to make some hard decisions. If a task is broken down into large number of very small activities, these can be carried out independently. Thus, it becomes possible to develop the product faster (with the help of additional manpower). Therefore, to be able to complete a project in the least amount of time, the manager needs to break large tasks into smaller ones, expecting to find more parallelism. However, it is not useful to subdivide tasks into units which take less than a week or two to execute. Very fine subdivision means that a disproportionate amount of time must be spent on preparing and revising various charts.

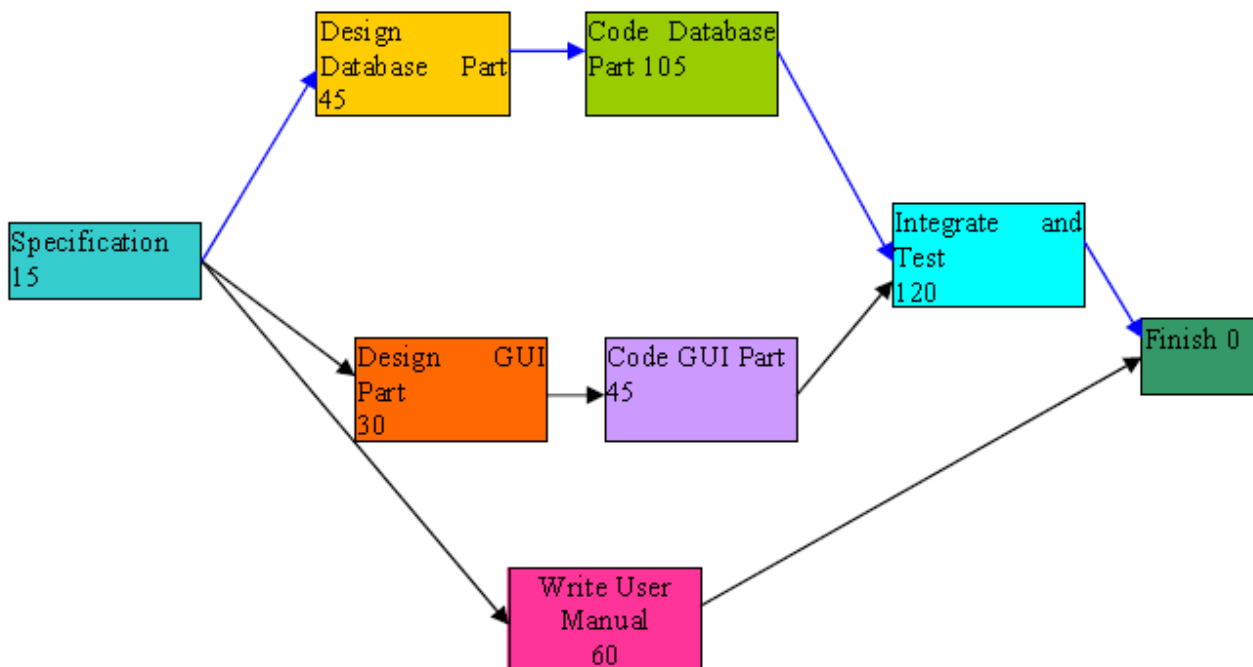


Work breakdown structure of an MIS problem

Activity networks and critical path method

WBS representation of a project is transformed into an activity network by representing activities identified in WBS along with their interdependencies. An activity network shows the different activities making up a project, their estimated durations, and interdependencies as shown in the figure. Each activity is represented by a rectangular node and the duration of the activity is shown alongside each task.

Managers can estimate the time durations for the different tasks in several ways. One possibility is that they can empirically assign durations to different tasks. This however is not a good idea, because software engineers often resent such unilateral decisions. A possible alternative is to let engineer himself estimate the time for an activity he can assigned to. However, some managers prefer to estimate the time for various activities themselves. Many managers believe that an aggressive schedule motivates the engineers to do a better and faster job. However, careful experiments have shown that unrealistically aggressive schedules not only cause engineers to compromise on intangible quality aspects, but also are a cause for schedule delays. A good way to achieve accurately in estimation of the task durations without creating undue schedule pressures is to have people set their own schedules.



Activity network representation of the MIS problem

Critical Path Method (CPM)

From the activity network representation following analysis can be made. The minimum time (MT) to complete the project is the maximum of all paths from start to finish.

The earliest start (ES) time of a task is the maximum of all paths from the start to the task.

The latest start time is the difference between MT and the maximum of all paths from this task to the finish.

The earliest finish time (EF) of a task is the sum of the earliest start time of the task and the duration of the task.

The latest finish (LF) time of a task can be obtained by subtracting maximum of all paths from this task to finish from MT.

The slack time (ST) is $LS - EF$ and equivalently can be written as $LF - EF$. The slack time (or float time) is the total time that a task may be delayed before it will affect the end time of the project. The slack time indicates the "flexibility" in starting and completion of tasks. A critical task is one with a zero slack time. A path from the start node to the finish node containing only critical tasks is called a critical path. These parameters for different tasks for the MIS problem are shown in the following table.

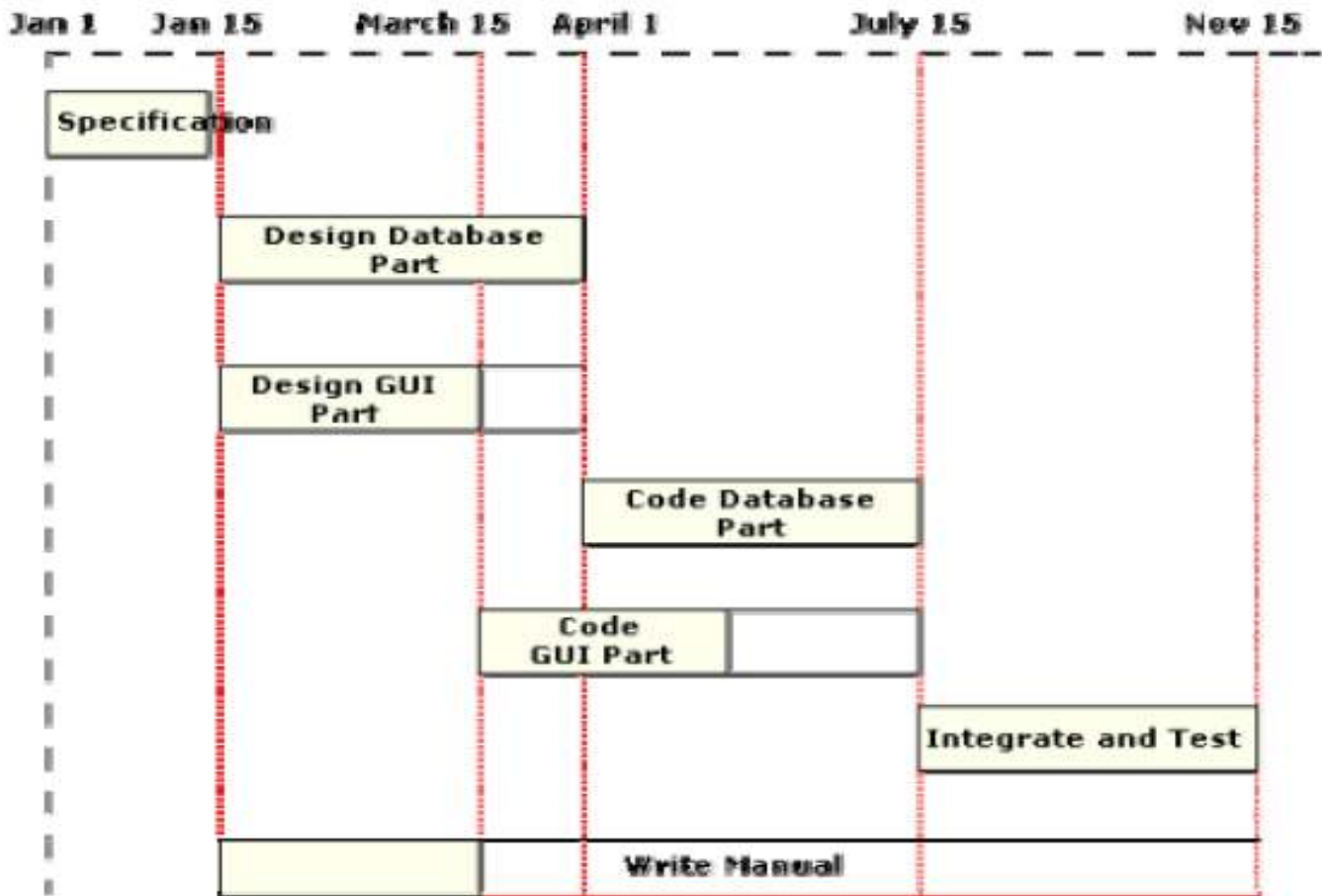
Task	ES	EF	LS	LF	ST
Specification	0	15	0	15	0
Design database	15	60	15	60	0
Design GUI part	15	45	90	120	75
Code database	60	165	60	165	0
Code GUI part	45	90	120	165	75
Integrate and test	165	285	165	285	0
Write user manual	15	75	225	285	210

The critical paths are all the paths whose duration equals MT. The critical path in figure is shown with a blue arrow.

Gantt chart

Gantt charts are mainly used to allocate resources to activities. The resources allocated to activities include staff, hardware, and software. Gantt charts (named after its developer Henry Gantt) are useful for resource planning. A Gantt chart is a special type of bar chart where each bar represents an activity. The bars are drawn along a time line. The length of each bar is proportional to the duration of time planned for the corresponding activity.

Gantt charts used in software project management are actually an enhanced version of the standard Gantt charts. In the Gantt charts used for software project management, each bar consists of a white part and a shaded part. The shaded part of the bar shows the length of time each task is estimated to take. The white part shows the slack time, that is, the latest time by which a task must be finished. A Gantt chart representation for the MIS problem of previous figure is shown in the figure below.

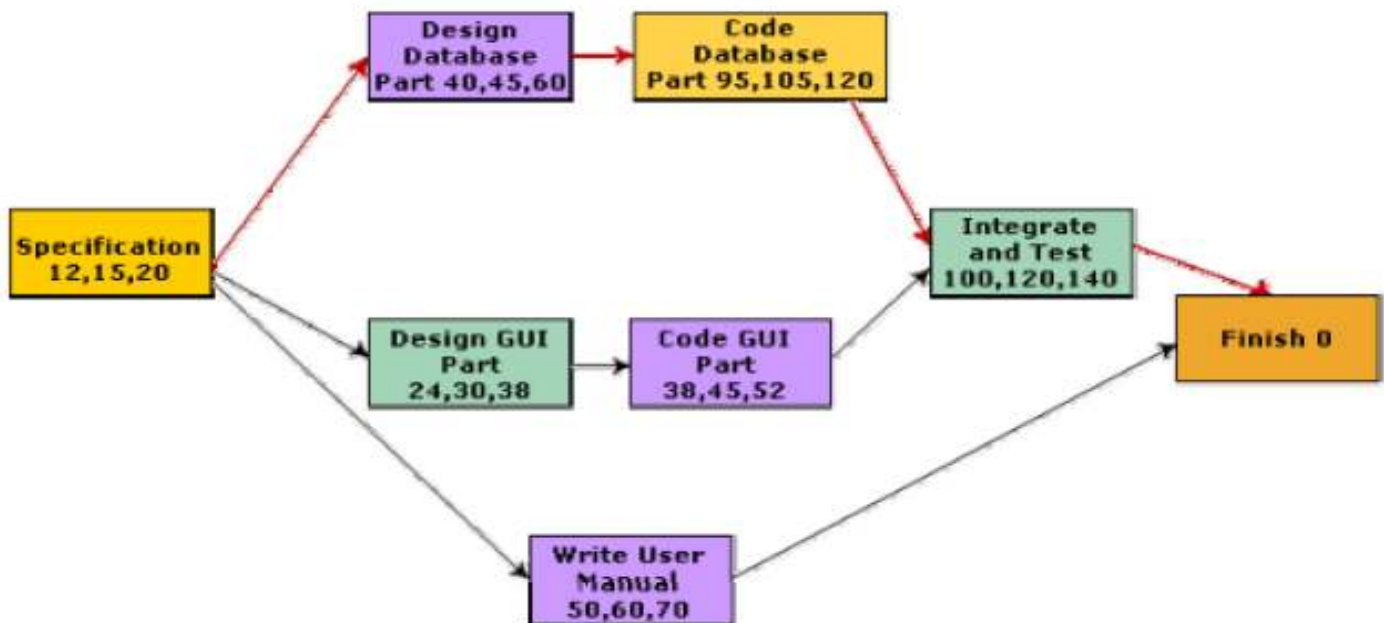


Gantt chart representation of the MIS problem

PERT chart

PERT (Project Evaluation and Review Technique) charts consist of a network of boxes and arrows. The boxes represent activities and the arrows represent task dependencies. PERT chart represents the statistical variations in the project estimates assuming a normal distribution. Thus, in a PERT chart instead of making a single estimate for each task, pessimistic, likely, and optimistic estimates are made. The boxes of PERT charts are usually annotated with the pessimistic, likely, and optimistic estimates for every task. Since all possible completion times between the minimum and maximum duration for every task has to be considered, there are not one but many critical paths, depending on the permutations of the estimates for each task. This makes critical path analysis in PERT charts very complex. A critical path in a PERT chart is shown by using thicker arrows. The PERT chart representation of the MIS problem of previous figure is shown in figure below. PERT charts are a more sophisticated form of activity chart. In activity diagrams only the estimated task durations are represented. Since, the actual durations might vary from the estimated durations, the utility of the activity diagrams are limited.

Gantt chart representation of a project schedule is helpful in planning the utilization of resources, while PERT chart is useful for monitoring the timely progress of activities. Also, it is easier to identify parallel activities in a project using a PERT chart. Project managers need to identify the parallel activities in a project for assignment to different engineers.



PERT chart representation of the MIS problem